Industrial Strength Computational Science for DOE: A V&V Foundation

Tim Trucano

Computational Physics Research and Development Department

Sandia National Laboratories

Albuquerque, NM 87185

Phone: 844-8812, FAX: 844-0918

Email: tgtruca@sandia.gov

Symposium on Validation and Verification of Computational Mechanics Codes, California Institute of Technology, December 9-11, 1998.

Computational Physics R&D Homepage:

http://sherpa.sandia.gov/9231home/compphys-frame.html

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

A little understanding goes a long way...

“In terms of the significance which is disclosed in understanding the world, concernful Being-alongside the ready-to-hand gives itself to understand whatever involvement that which is encountered can have."

Martin Heidegger, <u>Being and Time</u>

I want to address “technology thrusts” for V&V:

- Software quality assurance
- Software engineering techniques
- Sensitivity analysis
- Uncertainty analysis
- Where are we going?

There are <u>two assumptions</u> that underlie this talk:

- The goal is to provide information, not codes (don’t confuse means with ends!)

- The focus of our code development is engineered software product rather than science

ASCI program managers are not the only ones that say controversial things about computational science.

"If the capabilities of computers continue to increase as they have in the past, the relative roles of experiment and computation in providing aerodynamic flow simulations will undergo profound changes."

D. R. Chapman, H. Mark, and M. W. Pirtle (1975), "Computers vs Wind Tunnels for Aerodynamic Flow Simulations," Astronautics and Aeronautics, April, 22-35.

They stressed that economics was a major component in pressure on wind tunnel facilities. (Does this sound familiar?)

The authors quantitatively estimated that ~ 400 GFlops would do the trick. Duh...

So exactly what is <u>Verification</u> and <u>Validation</u>?

Physics code development:

Verification: “Are we solving the equations correctly”?

Validation: “Are we solving the correct equations”?

V&V must be coupled with code “accreditation” or “certification”.

The world is moving toward increased formal validation content in computational science and its application. Here are some, but not all, benchmarks (Oberkampf):

Society for Computer Modeling (1979)

IEEE (1984)

ASME (beginning 1986 and still evolving 1993)

American Nuclear Society (1987)

Military Operations Research Society (?)

Defense Modeling and Simulation Organization (1994)

AIAA (1994, 1998)

DOE DP [ASCI] (1998); other formal SQA activities prior to this date

ISO standards (?)

There is a lot of information out there.

Pat Roache has written the book: P. J. Roache Verification and Validation in Computational Science and Engineering , Hermosa, 1998

Another interesting book is: P. L. Knepell and D. C. Arangno, Simulation Validation - A Confidence Assessment Methodology, IEEE Computer Society, 1993

Here is a smattering of bibliographies:

W. O. Oberkampf (1998), SAND98-2041, several hundred

O. Balci and R. G. Sargent (1984), Simuletter, Vol. 15, No. 3, several hundred

O. Balci, date unknown, bibliography available at the DMSO Web Site

D. S. (Steve) Stevenson (1998), unpublished, ~ 150 references

T. G. Trucano (1998), unpublished, ~ 200 references

What are key characteristics of “multiphysics” codes?

Multiphysics often implies general purpose

General purpose implies that users are a huge component in correct application of the code

Multiphysics also often implies research models are present

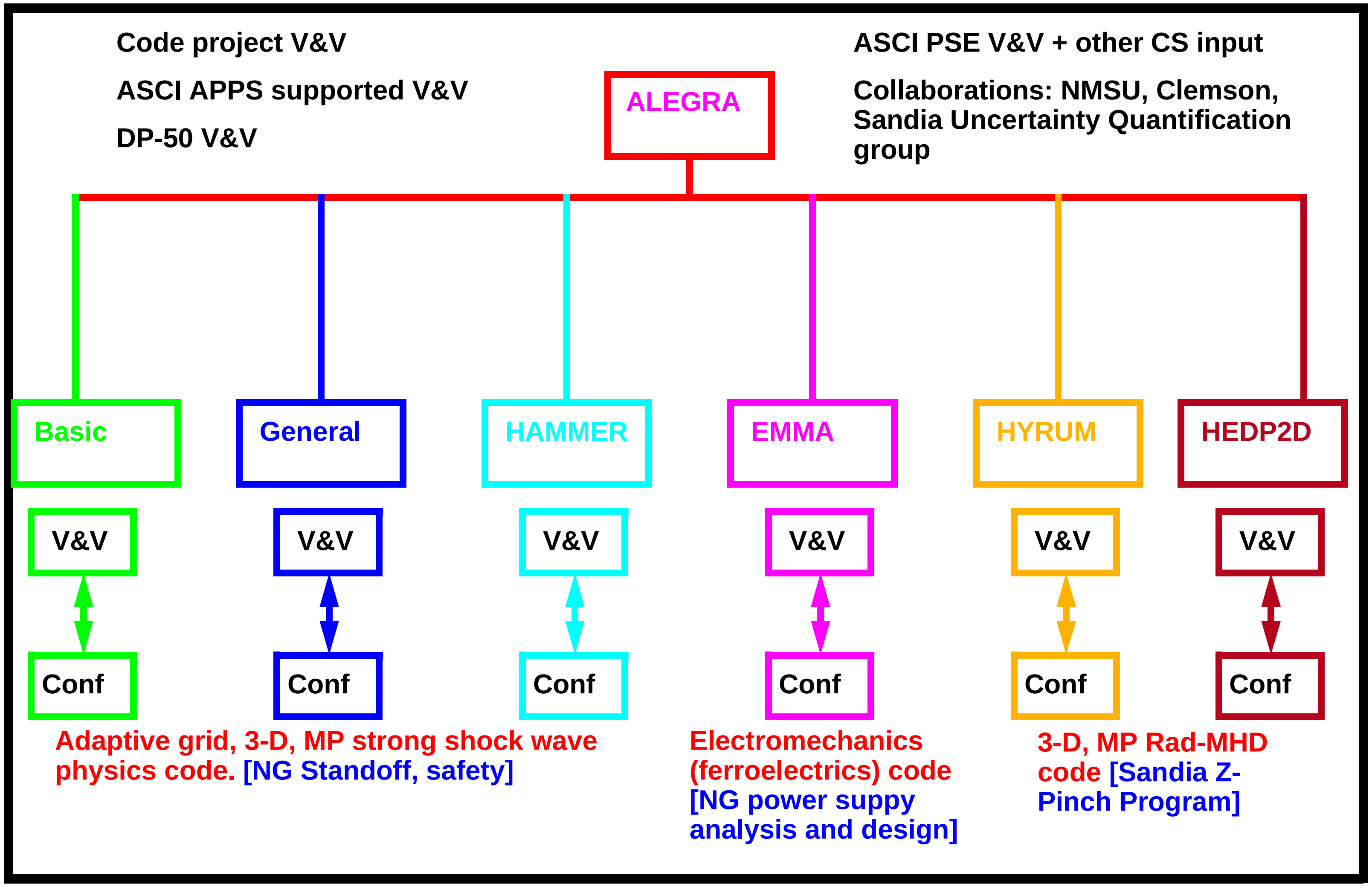
CTH:

- 3-D multimaterial Eulerian shock wave physics code
- Fortran (500,000 lines) + MPI + bits of C, etc.
- Hundreds of users (around the country)

ALEGRA:

- 3-D multimaterial Arbitrary Lagrangian Eulerian MHD shock wave physics code
- C++ (150000 lines) + MPI + C + Fortran, etc. (1,000,000 total including special libraries)
- Under development

ALEGRA is a multiphysics code:


Code project V&V
ASCI APPS supported V&V
DP-50 V&V
ALEGRA
ASCI PSE V&V + other CS input
Collaborations: NMSU, Clemson, Sandia Uncertainty Quantification group
Basic
General
HAMMER
EMMA
HYRUM
HEDP2D
V&V
V&V
V&V
V&V
V&V
V&V
Conf
Conf
Conf
Conf
Conf
Conf
Adaptive grid, 3-D, MP strong shock wave physics code. [NG Standoff, safety]
Electromechanics (ferroelectrics) code [NG power suppy analysis and design]
3-D, MP Rad-MHD code [Sandia Z-Pinch Program]

Software quality - is this an oxymoron?

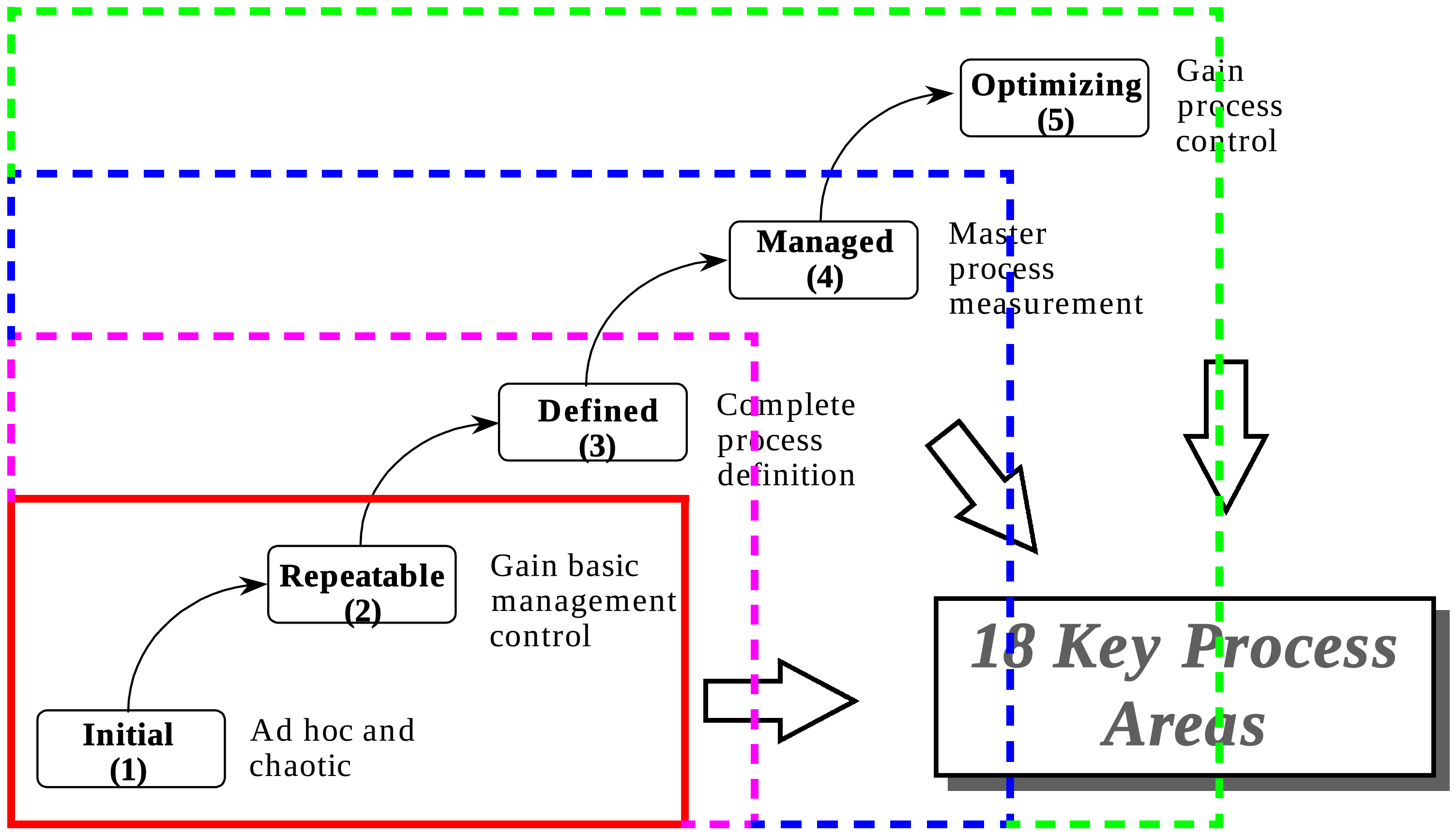
How much do you believe my premise that we are delivering software product?

> **"...I'm here to make sure that they [V&V Program] don't screw us up...", ASCI code development program person...**

SQA with ASCI code development - one view:

> **Good application of the principles to begin with**
>
> **Lax review approach during the software life cycle**
>
> **No metrics of the software life cycle process**
>
> **"Our guidance is that we expect a formal SQA program to be used with ASCI", DOE-AL person**

The Capability Maturity Model has the highest level of associated interest.



Delivering computational science as product requires software engineering.

What software engineering principles and techniques are applicable to large systems of floating point scientific code?

> **"...the author cannot find any long term empirical study on the efficacy of the various software engineering techniques...",** D. E. Stevenson (1997), "1001 Reasons for not Proving Programs Correct: A Survey," Clemson preprint.

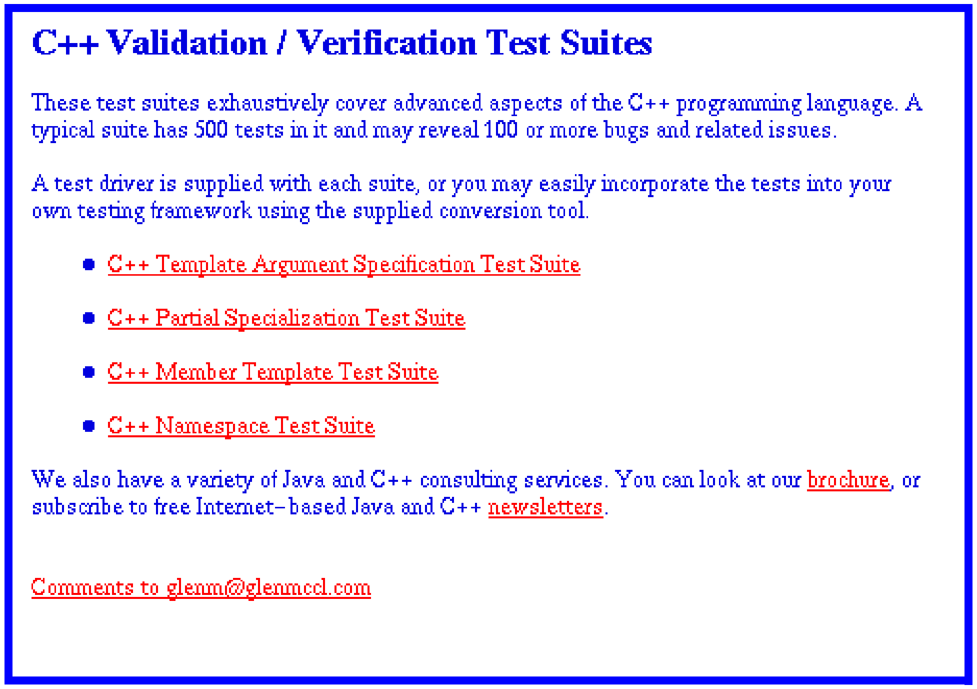
Many recommendations are called but less are chosen:

- Configuration management systems and rules **
- Program development environments (is the ideal to have no human write any code, but only specs, equations, etc?) *
- Coding standards*
- Formal software inspections*
- Mandatory regression testing **
- ???

**** - ALEGRA doing it; * - ALEGRA doing it half-baked**

If only it were true that ...

Verification fits this model relatively well, except...



C++ Validation / Verification Test Suites

These test suites exhaustively cover advanced aspects of the C++ programming language. A typical suite has 500 tests in it and may reveal 100 or more bugs and related issues.

A test driver is supplied with each suite, or you may easily incorporate the tests into your own testing framework using the supplied conversion tool.

- C++ Template Argument Specification Test Suite
- C++ Partial Specialization Test Suite
- C++ Member Template Test Suite
- C++ Namespace Test Suite

We also have a variety of Java and C++ consulting services. You can look at our brochure, or subscribe to free Internet-based Java and C++ newsletters.

Comments to glenm@glenmcd.com

this still doesn't "prove" that the code implementation is "correct."

"Thus a new component of the real has just appeared to us - non-being.", Jean-Paul Sartre, Being and Nothingness (The last code inspection we did found a bug that passed every benchmark test we had.)

The “God” of testing:

B. Beizer, Software Testing Techniques, International Thomson Computer Press, 1990

For example (software):

- Flowgraphs and path testing
- Transaction-flow testing
- Data-flow testing
- Domain testing
- Syntax testing
- Logic-based testing
- Transition testing

For example (novel):

- Spelling
- Grammar
- Punctuation
- Paragraph structure
- Syntax
- Logic

At the end of all of this you still don’t have a novel.

Verification for a multi-physics code is more than code testing: the meta-issues are what kill you!

What tests?

How many of them?

What is the benefit?

What is the cost?

What are they telling you?

What are they not telling you?

What are the metrics?

But you'd better worry about testing a lot! - "The T Experiments"

L. Hatton (1997), "The T Experiments: Errors in Scientific Software," Computational Science and Engineering, Vol. 4, No. 2, 1997, 27-38

"The results were horrifying." - Roache

Here are the lessons to me:

- **Well designed static testing is critical.**
- **You need *tools* to do this.**
- **You need *metrics* to do this.**
- **The proper language to understand what Hatton is saying is *reliability*, not physics.**
- **For those who must believe that code validation can be accomplished through code comparisons (I don't!!!), Hatton really illustrates how to do a code comparison.**

Dynamic Testing 101 - Does everybody know what regression testing is?

Regression testing tests implementation stability, not correctness. It is strictly speaking a software engineering practice, not a verification activity per se.

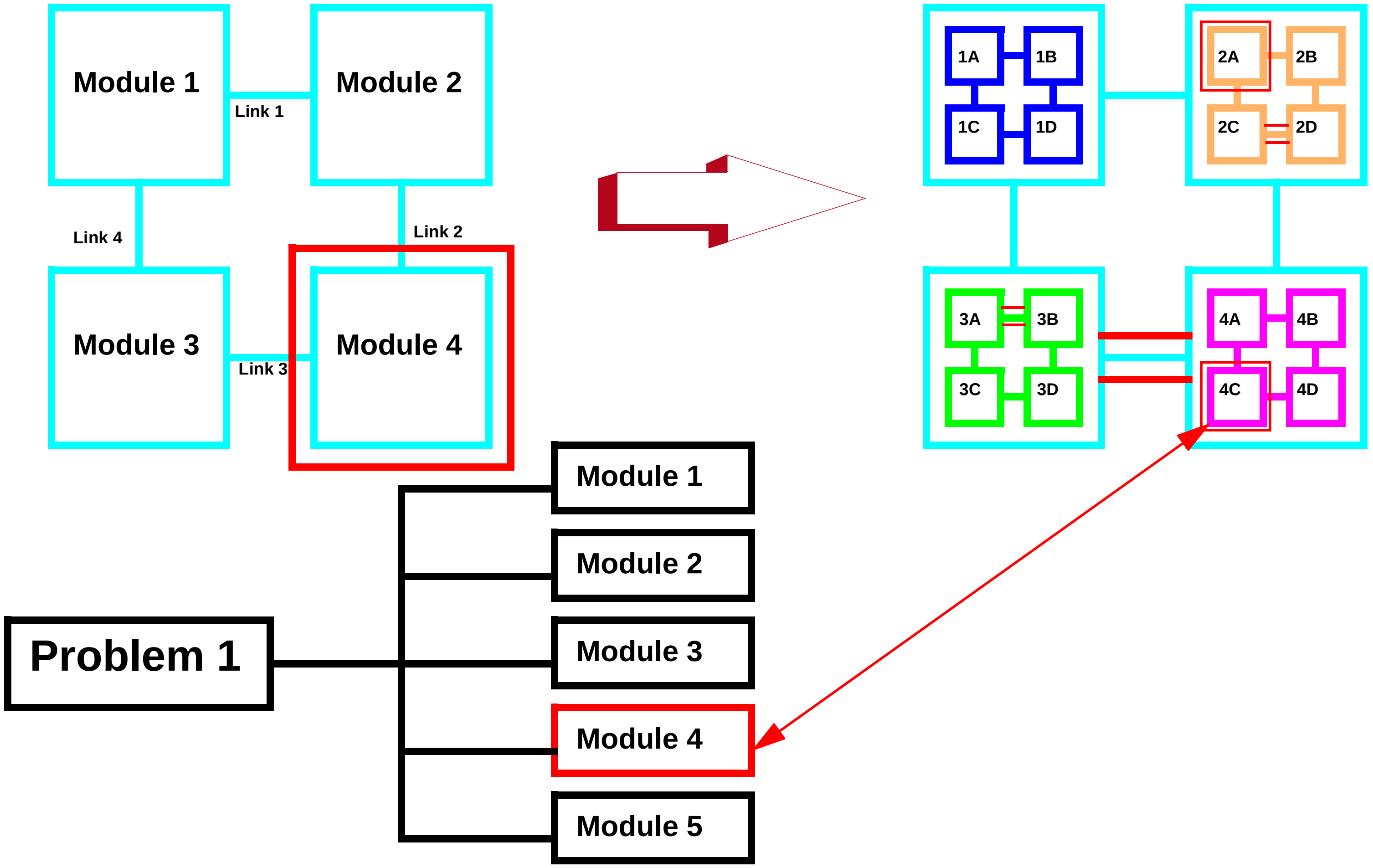
- **We do it and it leads to much interesting insight.**
- **The regression test set covers less than 50% of the functioning software (hmmm).**
- **The appropriate metric is that we spend much less time fixing bad checkins than we did before it - but they still creep in even in parts of the code covered by the test suite.**
- **Serves as the kernel for a larger verification test suite.**

Dynamic Testing 201 - Component testing and coverage.

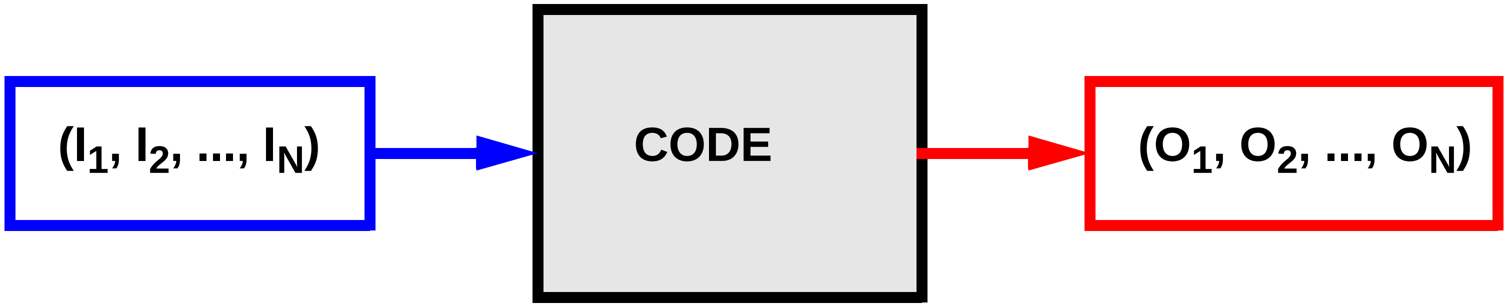
What level of code component granularity is most effective for techniques like regression testing?

- **We need to assess both module and path coverage for specific problems. This requires tools.**
- **We can't test every single "module" and path at the lowest level. We are driven to larger modules, which implicitly contain more paths.**
- **What is an appropriate module size for efficient testing (so it does not take two weeks of running before you can do a code check in?)**
- **What is an appropriate measure of module importance? Of path importance?**
- **Factor in the probability that ASCI hardware and OS environments will likely cause regression test failures themselves.**
- **ALEGRA is currently covered on the order of 40% by its regression test suite.**

Dynamic testing 301 - Links between test problems and code modules are needed for component testing.



Dynamic Testing 401 - What is sensitivity analysis and why do I care?



$$\frac{\partial O_i}{\partial I_j}$$

$$\frac{\partial^2 O_i}{\partial I_j \partial I_k}$$

Question: what input parameters are most important for a given application?

Dynamic Testing 401 - Sensitivity analysis can be used to determine the most important modules for verification.

Sensitivity analysis can be used to refine the regression test suite, as well as the more intensive verification problem suites.

Coupled with suitable coverage tools, sensitivity analysis tools can suggest important paths as well as modules.

Sensitivity analysis can be used to develop measures of "importance."

How should sensitivity coefficients be calculated?

- Deterministic methods (brute force numerical differentiation of the output is not recommended)
- "Symbolic methods" - automatic differentiation (not yet applicable to us)
- Probabilistic methods (regression based, for example)

Dynamic Testing 501 - what about self-generated test problems?

Simple

Invert sources, say, to determine coefficients, then solve the resulting direct problem and confirm you match the source

Heuristic

Synthesize test problems from a sampling of user problems

Do this continuously?

Artificial intelligence?

Some more sophisticated approach to sensing code weaknesses and designing test problems to attack those weaknesses

Dynamic Testing xxx - Let’s not forget the hardware environment.

Think of two things:

> The generic supercomputer in the 100 TFlop to 1 PFlop range is likely to be a heterogeneous system
>
> We fail regression and benchmark testing a significant amount of time due to “hardware” issues (OS, libraries, general system issues, and general hardware instability)

Should every important calculation be preceded by a super-Paranoia-for-supercomputers (Stevenson) that insures that the compute environment on the machine is ready for that calculation?

> “How expensive is it?”, Unknown supercomputer user...

Do we need validation “science”?

"...we never (least of all in science) draw inferences from mere observational experience to the prediction of future events. Rather, each such inference is based upon observational experience...plus some universal theories..."

Karl Popper, Objective Knowledge

What is “uncertainty quantification” and why do I care?

The forward prediction problem:

Characterize the “input” uncertainty (stochastic, fuzzy, etc)

Propagate this uncertainty through the code

Characterize the resulting output uncertainty

Refine this characterization via comparison with data

Develop “code reliability” metrics and statements (need requirements)

(Most of this is not rocket science for an initial implementation.)

Now follow it with backward prediction:

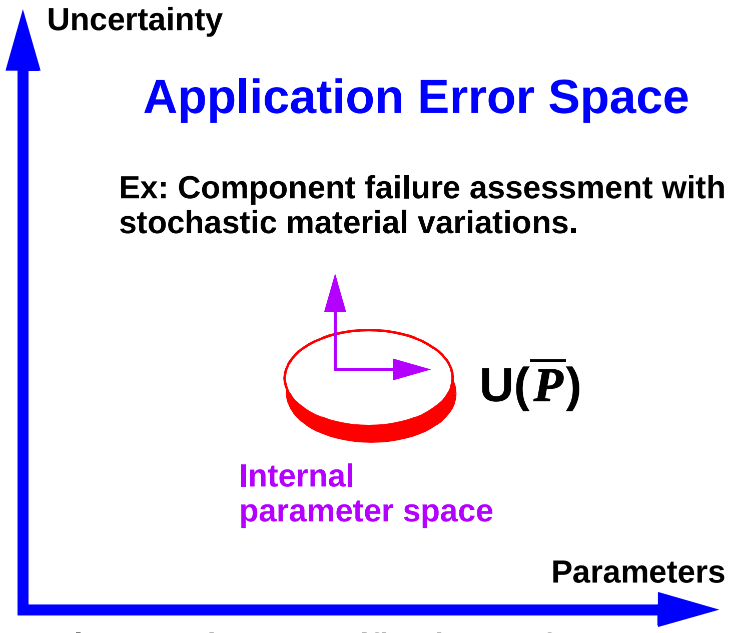
Reduce the code uncertainty via the output uncertainty characterization (Bayesian?).

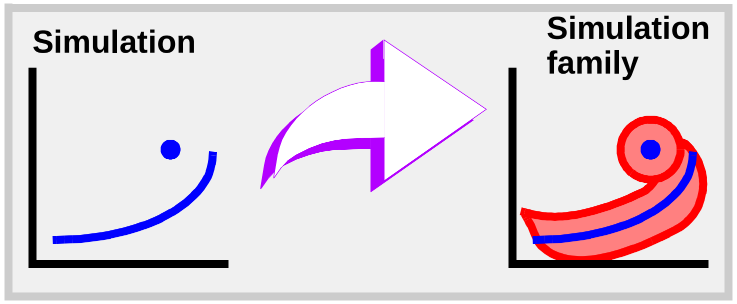
(This IS rocket science.)

Now optimize:

Perform forward/backward prediction sweeps to increase “code reliability” and guide new experiments.

Local validation begins with local uncertainty quantification, a high dimensional problem.





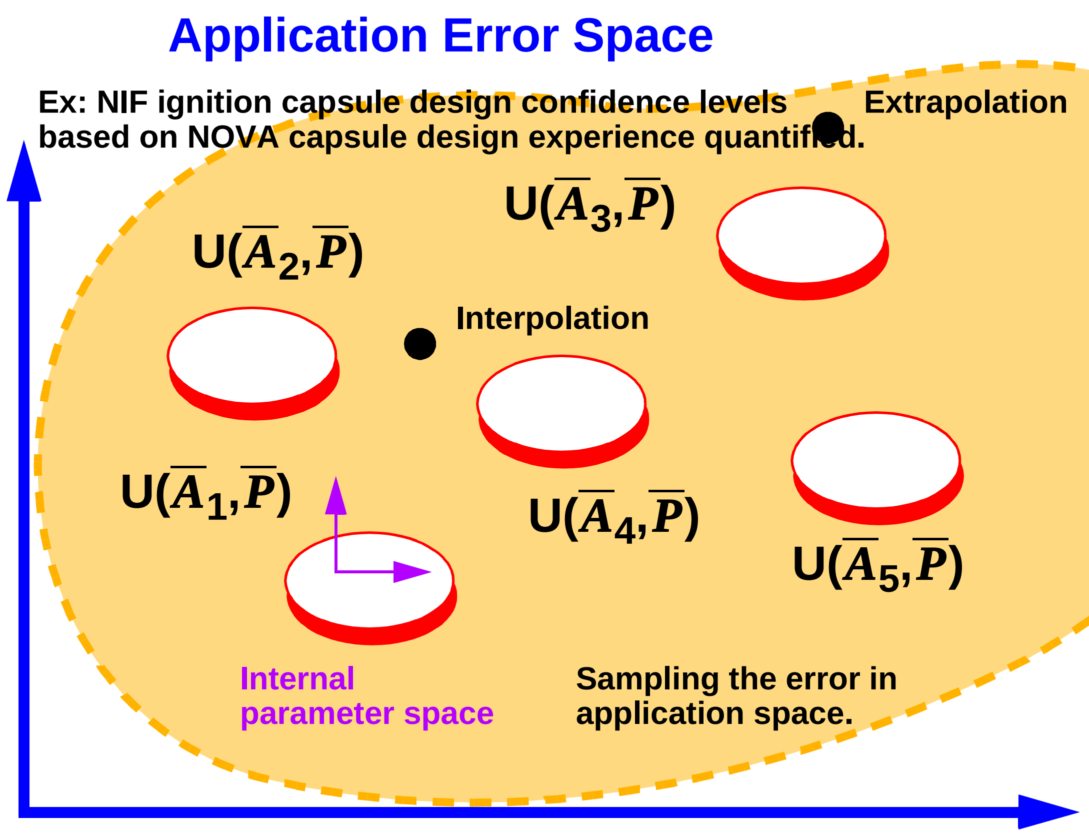
Local uncertainty quantification performs systematic studies of code uncertainty from stochastic treatments of parameter uncertainty. U is a random variable, $\overline{P}$ is a random vector.

Notice that how uncertainty is defined is an issue.

Alternatively, think of results being replaced by probability distributions of results. Then, do response surface methods, or other things, to characterize the result family. General issue is characterization

The number of parameters can be enormous - *parsimony* is hoped for.

Trying to assess global code uncertainty with stochastic inference leads to problems with very high dimensions ("internal" + "applications" parameters.


Application Error Space
Ex: NIF ignition capsule design confidence levels based on NOVA capsule design experience quantified.
Extrapolation
$U(\overline{A}_3,\overline{P})$
$U(\overline{A}_2,\overline{P})$
Interpolation
$U(\overline{A}_1,\overline{P})$
$U(\overline{A}_4,\overline{P})$
$U(\overline{A}_5,\overline{P})$
Internal parameter space
Sampling the error in application space.

Uncertainty has two components:

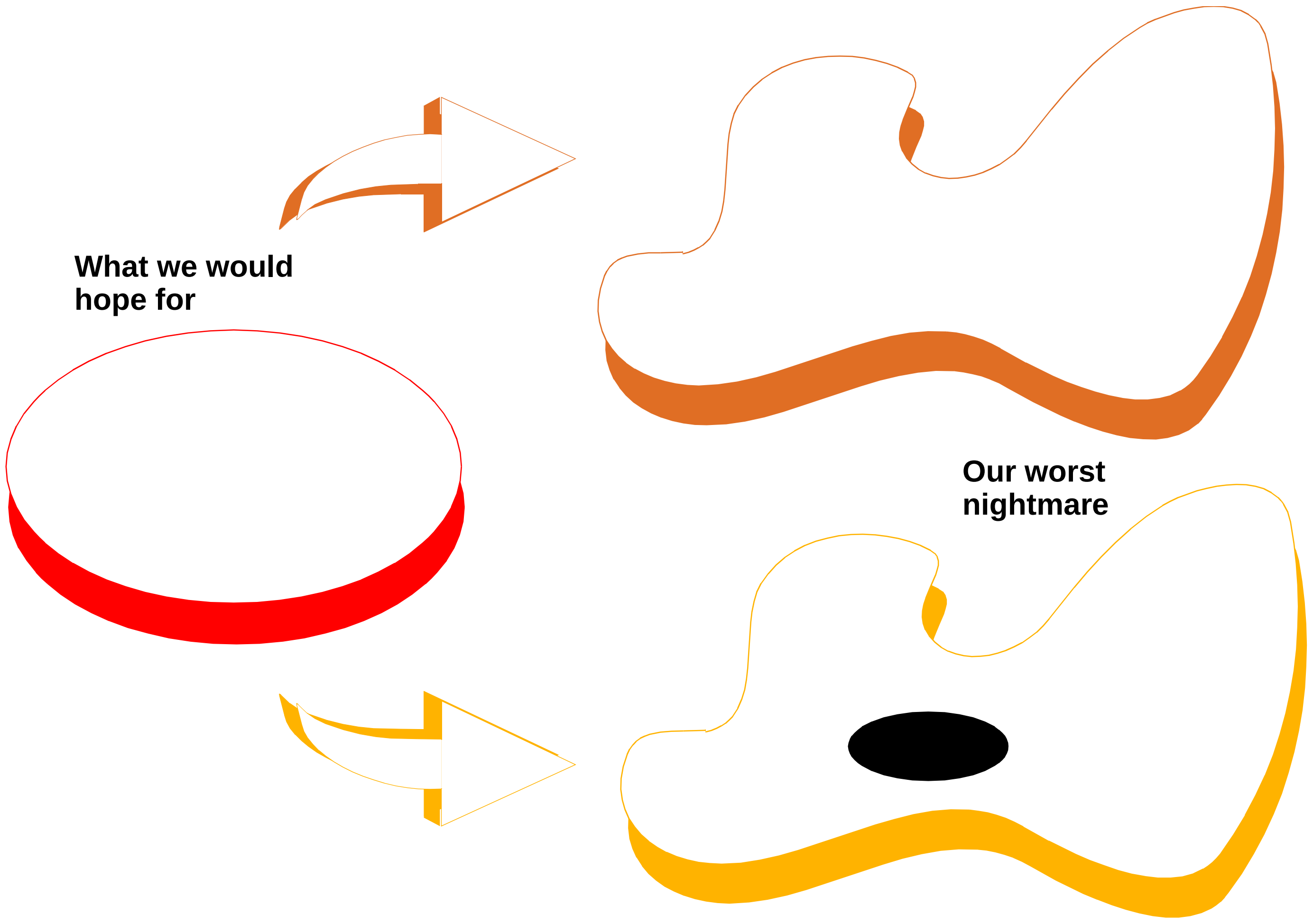
"Local" - uncertainty quantification and local data comparisons.

"Global" - system scale uncertainty quantification and global data comparisons.

Can we use spatial statistics methods (like kriging) to characterize $U(\overline{A},\overline{P})$?

Note that we have simplified the problem by assuming that the internal parameter functional dependence is constant over application space. Is this true?

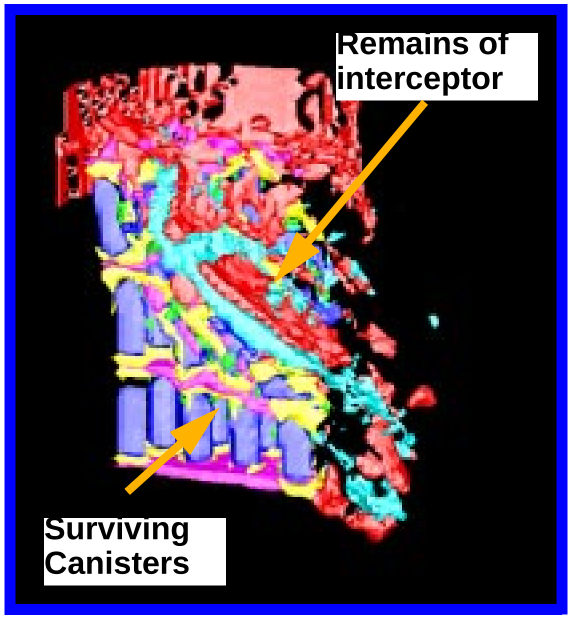
Note that "uncertainty" probably looks a lot more complex than suggested by the previous figures:

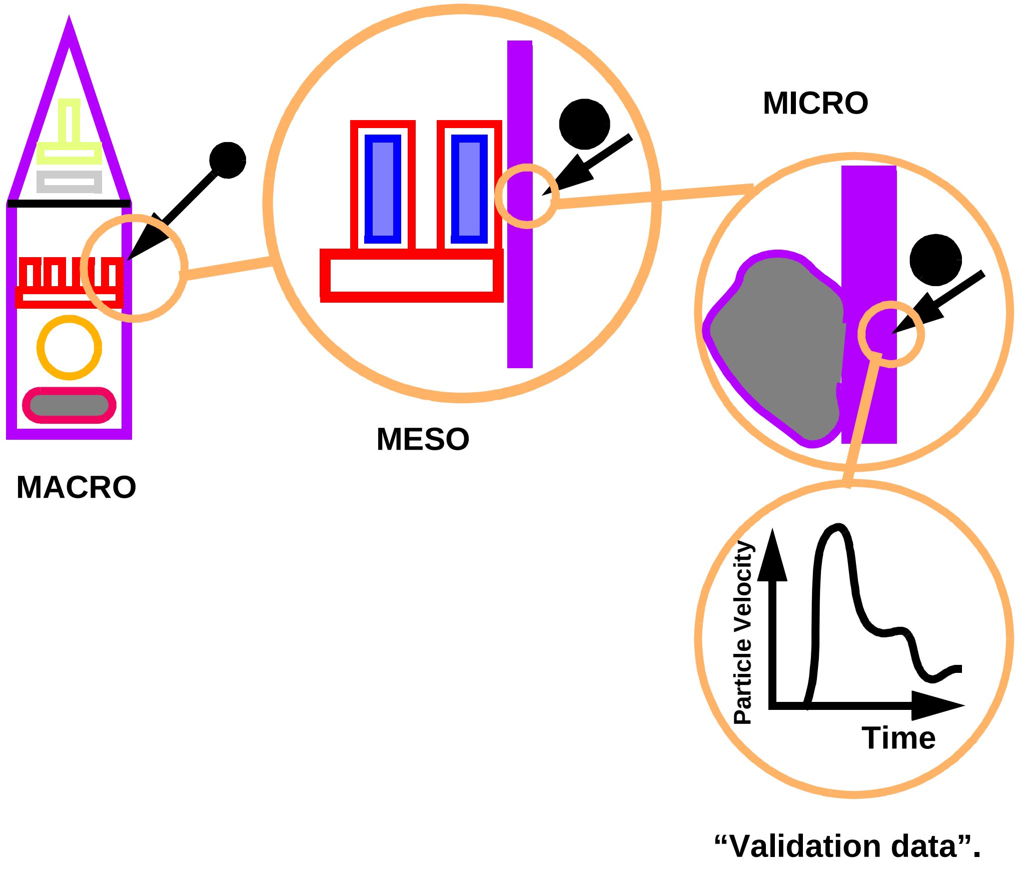


Application illustration - a hypevelocity impact example

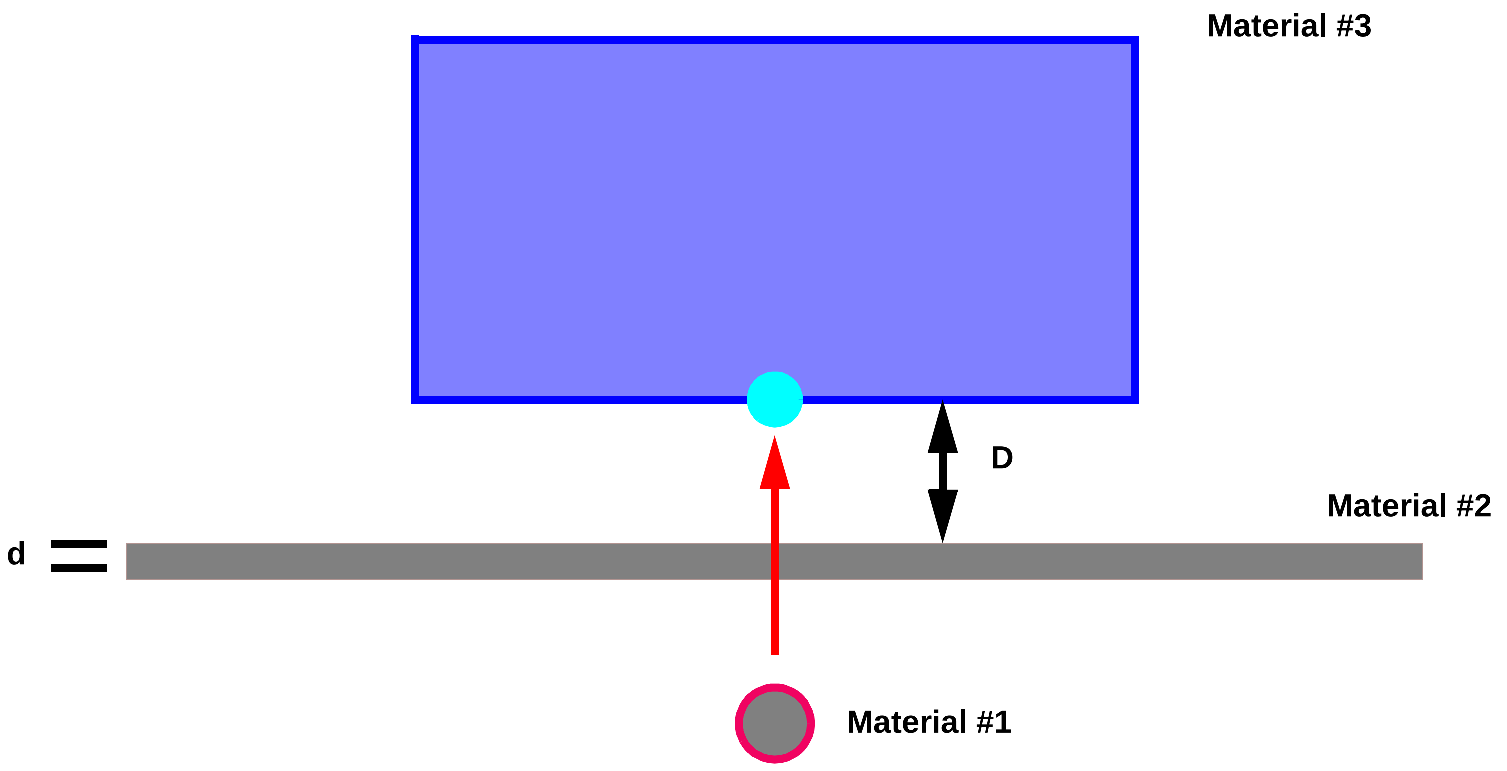
BMD simulations provide an excellent application for studying predictive complexity.

There are at least six stochastic parameters to begin with: the hit point and the engagement velocity.



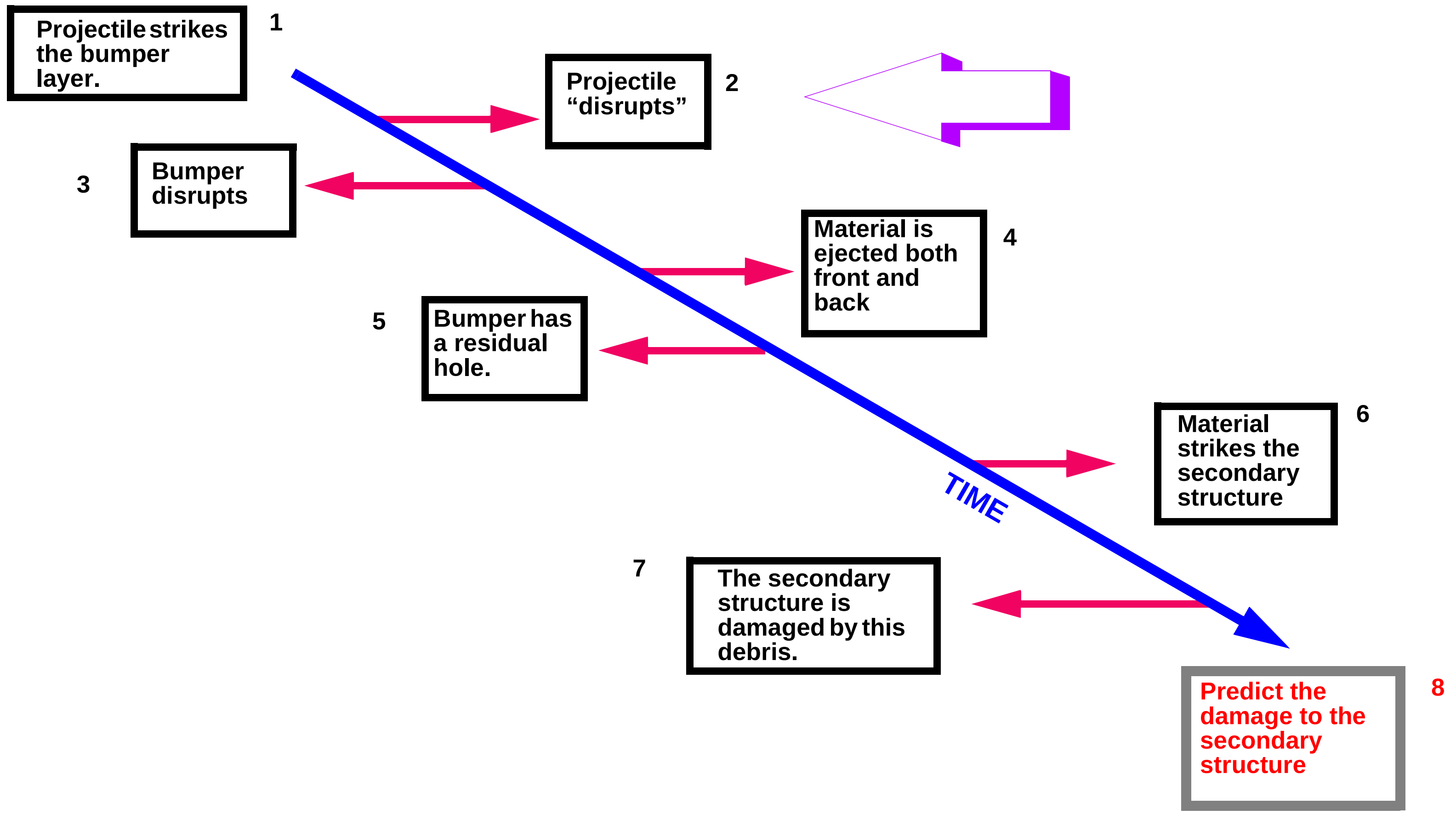


An experiment relevant to hypervelocity impact applications:

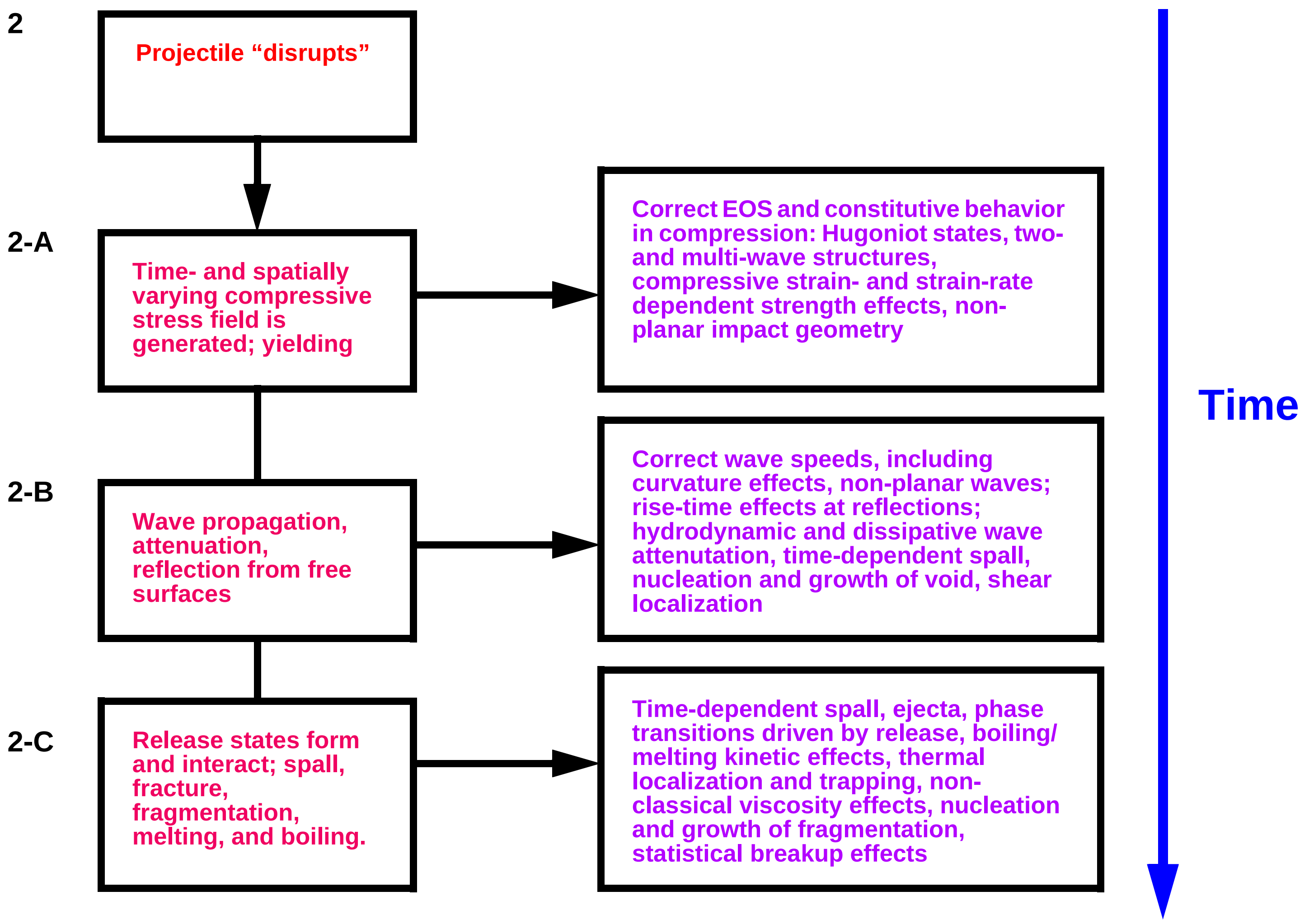


A simplified physical situation.

The levels of phenomenology in this experiment:


Projectile strikes the bumper layer.
1
Projectile "disrupts"
2
3
Bumper disrupts
Material is ejected both front and back
4
5
Bumper has a residual hole.
6
Material strikes the secondary structure
TIME
7
The secondary structure is damaged by this debris.
Predict the damage to the secondary structure
8

Further levels of phenomenology in this experiment:



2 Projectile “disrupts”

2-A Time- and spatially varying compressive stress field is generated; yielding

Correct EOS and constitutive behavior in compression: Hugoniot states, two- and multi-wave structures, compressive strain- and strain-rate dependent strength effects, non-planar impact geometry

2-B Wave propagation, attenuation, reflection from free surfaces

Correct wave speeds, including curvature effects, non-planar waves; rise-time effects at reflections; hydrodynamic and dissipative wave attenutation, time-dependent spall, nucleation and growth of void, shear localization

2-C Release states form and interact; spall, fracture, fragmentation, melting, and boiling.

Time-dependent spall, ejecta, phase transitions driven by release, boiling/ melting kinetic effects, thermal localization and trapping, non-classical viscosity effects, nucleation and growth of fragmentation, statistical breakup effects

What the parameter space looks like for this hypervelocity impact experiment:

"Experimental" parameters:

- **Impact location (2)**
- **Velocity vector (3)**
- **D, d (2)**
- **Projectile geometry (1)**
- **Materials (none)**

Total = 8

"Internal" parameters:

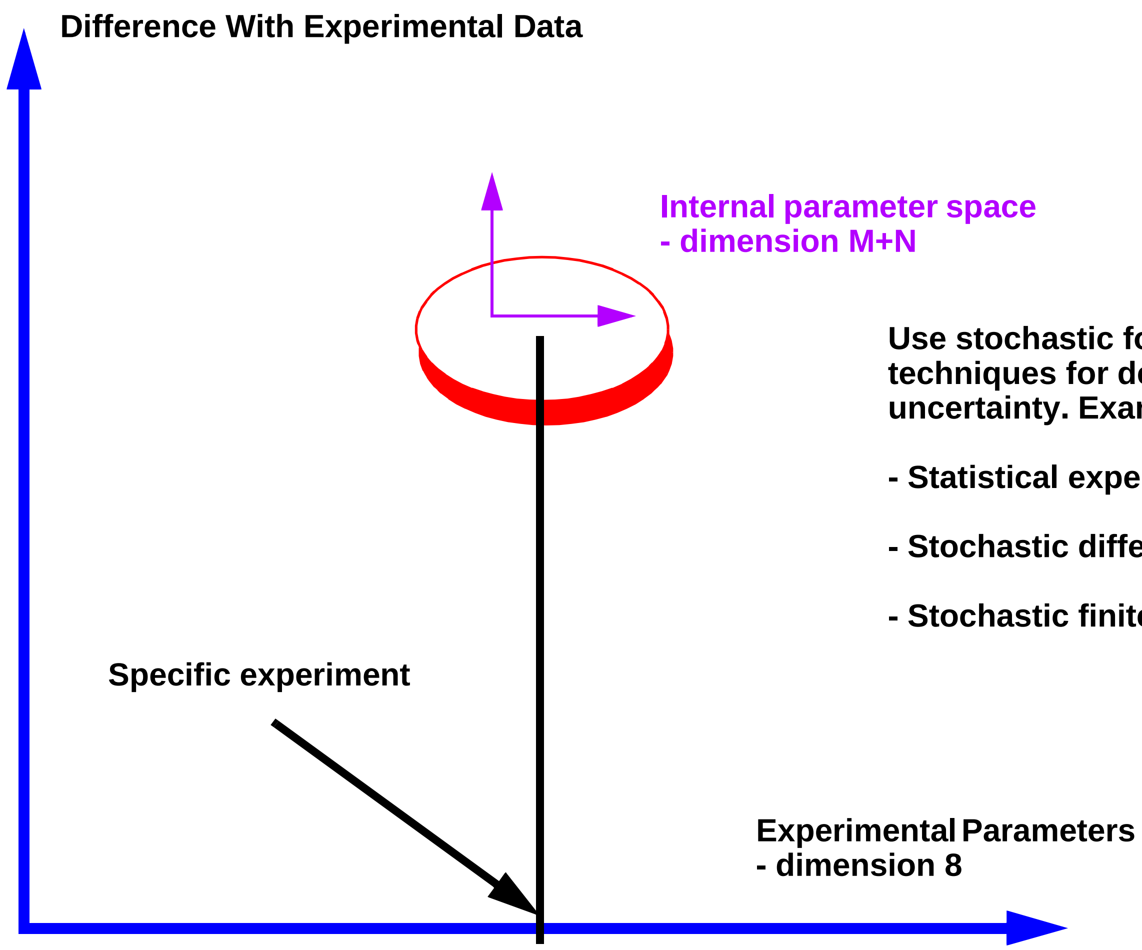
- **Hydrodynamic parameters (H)**
- **Material Models (M)**

Total = H + M

"Uncertainty" metrics:

- **Time resolved data in witness material (PVF gauges)**
- **Time- and spatially-resolved debris cloud data (radiagraphy)**
- **Target recovery and inspection**

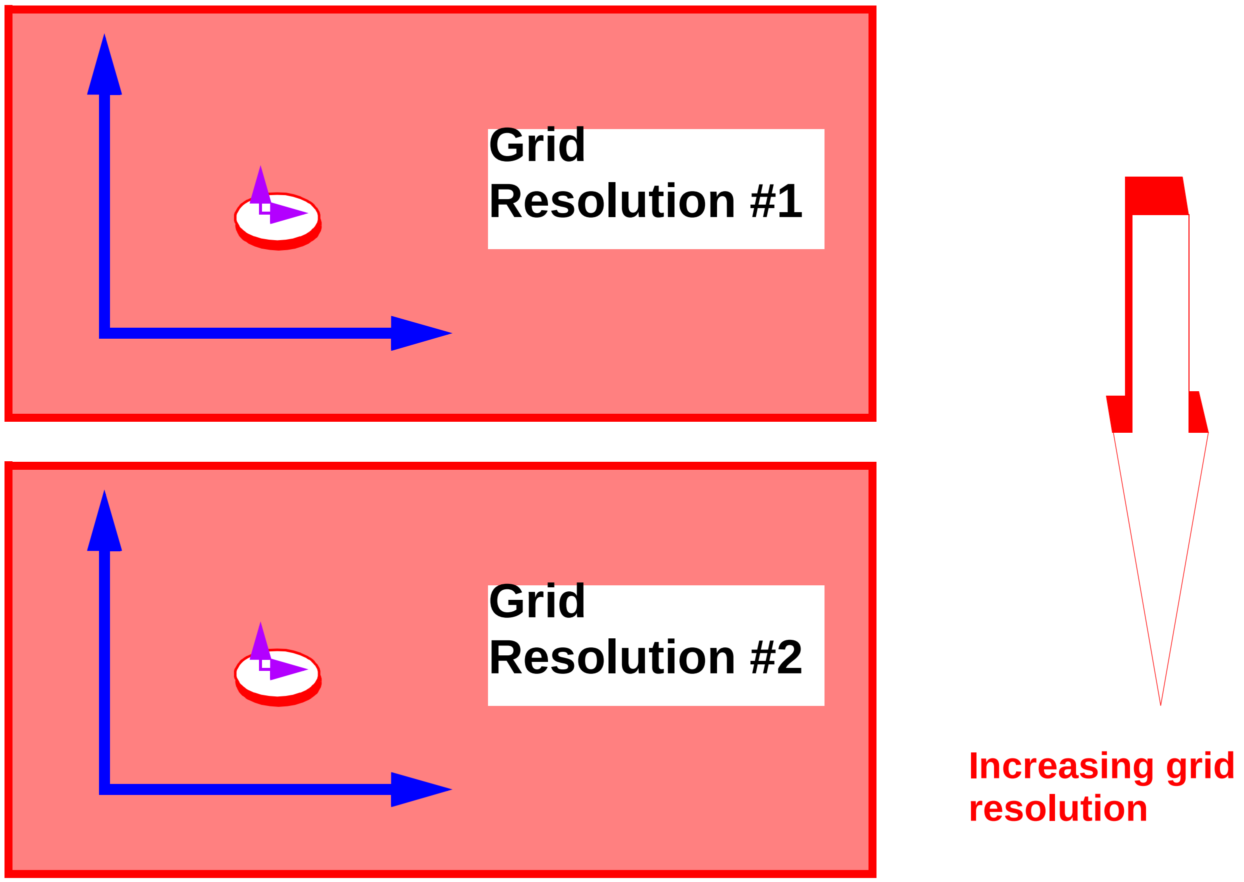
What the parameter vs uncertainty space looks like for this specific example:



Use stochastic forward propagation techniques for determining the “local” uncertainty. Examples include

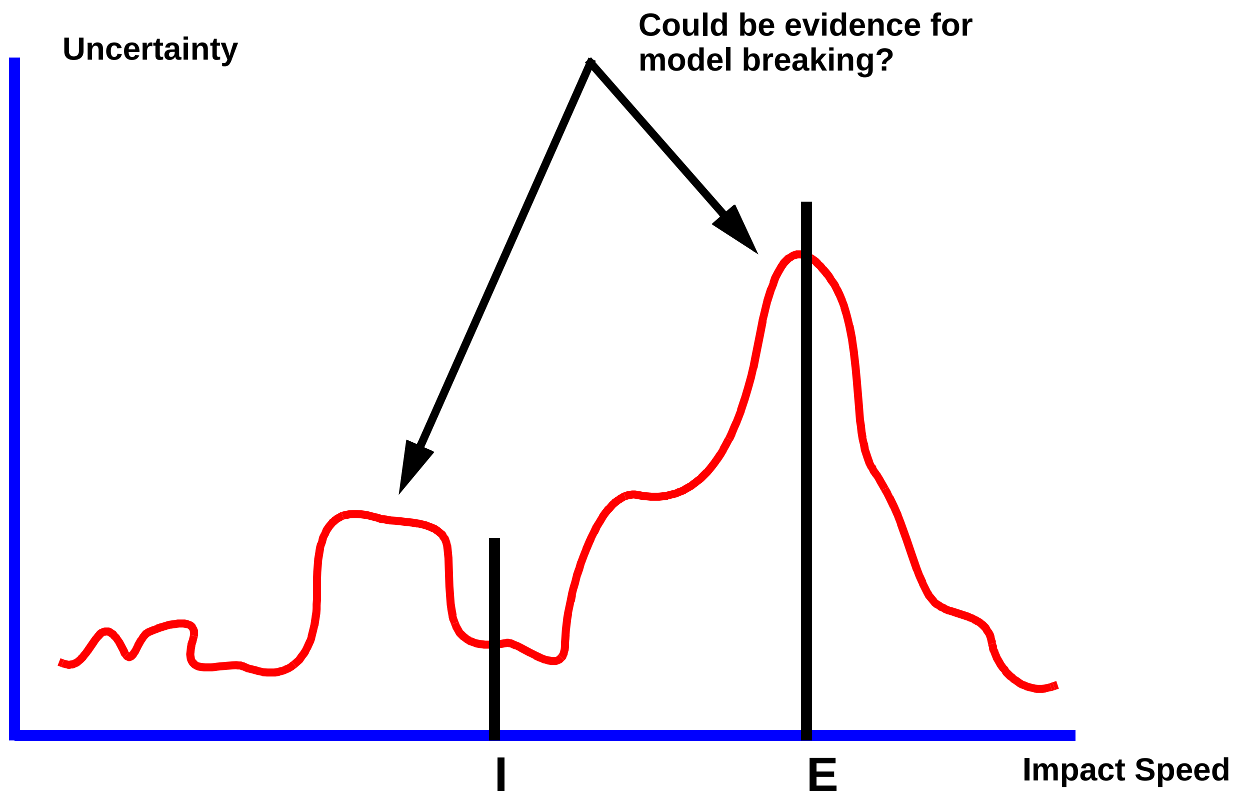
- Statistical experimental design (sampling)
- Stochastic differential equations
- Stochastic finite elements

Grid resolution in this approach:



Increasing grid resolution does not mean uniform refinement (ALE, adaptivity, geometry constraints). Algorithm parameters controlling dynamic grid resolution are included in the internal parameters.

Projecting the uncertainty will look like:



Some speculation and reasonable opportunities for research: Uncertainty as a spatial random field

I. Let the uncertainty be represented by a (M+H+8) dimensional spatial stochastic process

For example, we end up worrying about properties of something like the variogram:

$$\mathrm{var}[U(\vec{e}_j, \vec{i}_j) - U(\vec{e}_k, \vec{i}_k)] = 2\gamma((\vec{e}_j, \vec{i}_j) - (\vec{e}_k, \vec{i}_k))$$

Then, we can develop predictors for U at other points, such as the BLUP (Best Linear Unbiased Predictor):

$$\hat{U}(\vec{e}_0, \vec{i}_0) = \sum_{i=1}^{n} \lambda_i U(\vec{e}_j, \vec{i}_j)$$

You can get fancier. For example, does it make sense to seek predictors of the form:

$$U(\vec{e}, \vec{i}) = \mu(\vec{e}, \vec{i}) + S(\vec{e}, \vec{i}) + \varepsilon(\vec{e})$$

Here, μ is the mean of the random field, S contains fine structure about the nature of the random field, and ε is called measurement error. It is an additional but logical assumption that ε depends only on the experimental parameters.

See J. Sacks, W. J. Welch, T. J. Mitchell and H. P. Wynn (1989), "Design and Analysis of Computer Experiments," Statistical Science, Vol. 4, No. 4, 409-435; N. Cressie (1988), "Spatial Prediction and Ordinary Kriging," Mathematical Geology, Vol. 20, No. 4, 405-421.

Do we need another framework other than probability to do this?

What is the appropriate way to partition this random field among the experimental parameters and the internal parameters? Does this question even make sense?

What structure do we require on the projected random field $U(\overrightarrow{e_0}, \vec{i})$ to facilitate piecing together the various local uncertainties?

Speculation: Sensitivity Coefficients

II. Sensitivity studies define which of the H+M+8 parameters is most important. Probabilistic evaluation of the sensitivities is of interest.

Is parsimony really true?

Does the sensitivity structure projected onto the internal parameter space remain invariant as the experimental parameters alone vary? If no, does this imply model invalidity?

Does the sensitivity structure remain invariant over grid variations?

Don’t assume that the parameters are all uncorrelated. Then we need “interaction” coefficients.

The literature on sensitivity analysis is huge. See M. D. McKay (1995), “Evaluating Prediction Uncertainty,” Los Alamos Report, LA-12915-MS for one important approach.

Speculation: on model calibration

III. Some understanding of U should lead to improvement in the model. "Calibration" reduces U locally by optimizing the internal parameters.

How does the calibration vary with the experimental parameters?

A Bayesian approach can be applied to the formal study of improving model uncertainty in the presence of parameters derived via comparison with experimental data. See, for example, K. M. Hanson (1998), "A Framework for Assessing Uncertainties in Simulation Predictions," Los Alamos preprint.

Statistically rigorous comparisons between uncertain calculations and uncertain data with the intent of providing code validation are the subject of a recent review by R. G. Hills (1998), "Statistical Validation of Engineering and Scientific Models: Background," Sandia National Laboratories Contract Report.

This question also leads to the use of "surrogates" for studying parameter calibration, as well as other optimization questions associated with code uncertainty. Consider the important work A. J. Booker, et al (1998), "A Rigorous Framework for Optimization of Expensive Functions by Surrogates," Boeing Shared Services Group Report, SSGTECH-98-005.

Other papers that the reader might find of interest are D. D. Cox, J. Park, and C. E. Singer (1996), "A Statistical Method for Tuning a Computer Code to a Data Base," Rice University Department of Statistics Report 96-3 and M. B. Beck (1987), "Water Quality Modeling: A Review of the Analysis of Uncertainty," Water Resources Journal, Vol. 23, No. 8, 1393-1442.

Speculation: Those darn unknown unknowns

IV. Is there anyway to deal with “structural” uncertainty?

A Bayesian structure can be developed for considering structural uncertainty. See D. Draper (1995), “Assessment and Propagation of Model Uncertainty,” J. R. Statist. Soc. B, Vol. 57, No. 1, 45-97.

This involves developing posteriors via conditioning over the space of models, a rather hopeless endeavor on the face of it. Additional structure might make this more feasible.

Model uncertainty is often treated in multiphysics code through the introduction of tuning parameters. If a code (sub)model is built out of sub-submodels:

$$\boldsymbol{M} = \sum_j \boldsymbol{M}_j$$

Uncertainty about the overall model is then treated by modifying this equation to:

$$\boldsymbol{M} = \sum_j \alpha_j \boldsymbol{M}_j$$

Add (α_1, ..., α_m) to the parameter list and proceed as before.

Speculation: Additional issues

How do we treat “variability” in the assumed distributions on the parameters to do experimental design, sensitivity analysis, and forward propagation?

Is probability the canonical way to capture “uncertainty?”

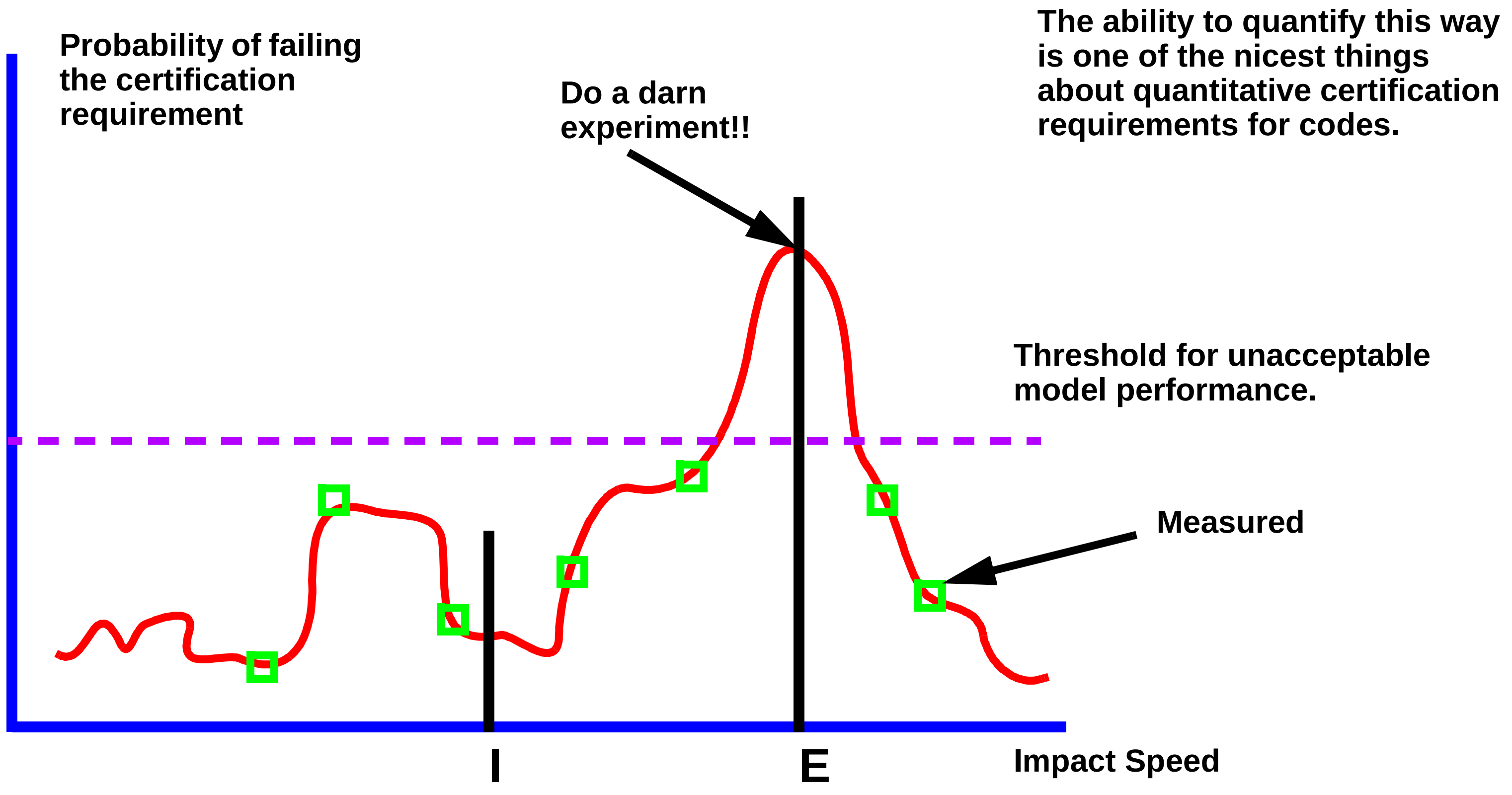
- **Is saying “I don’t know what the value of a parameter is” the same as placing a probability distribution on it?**

We need to design validation processes to attack the weak points of codes.

"...the theorist is interested in explanation as such, that is to say, in testable explanatory theories: applications and predictions interest him only for theoretical reasons - because they can be used as tests of theories..."

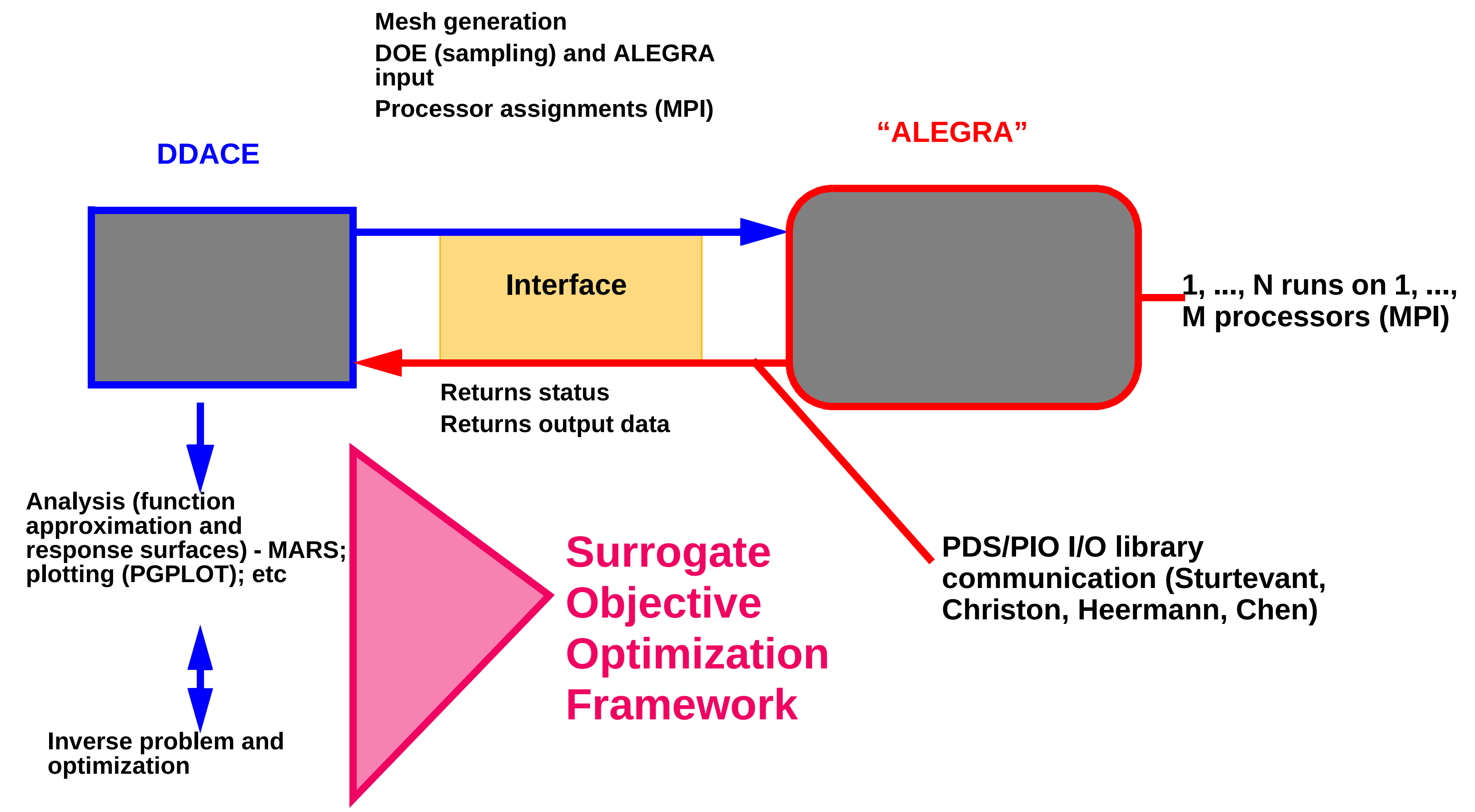
Karl Popper, The Logic of Scientific Discovery

Speculation: “Certification” leads to quantitative “reliability” analysis.



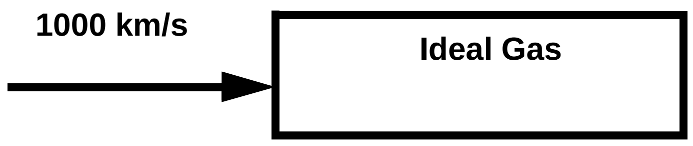
See D. G. Robinson (1998), “A Survey of Probabilistic Methods Used in Reliability, Risk, and Uncertainty Analysis: Analytical Techniques I,” Sandia National Labs Report SAND98-1189.

DDACE (DOOMSDACE: Distributed Object-Oriented Software with Multiple Samplings for the Design and Analysis of Computer Experiments)

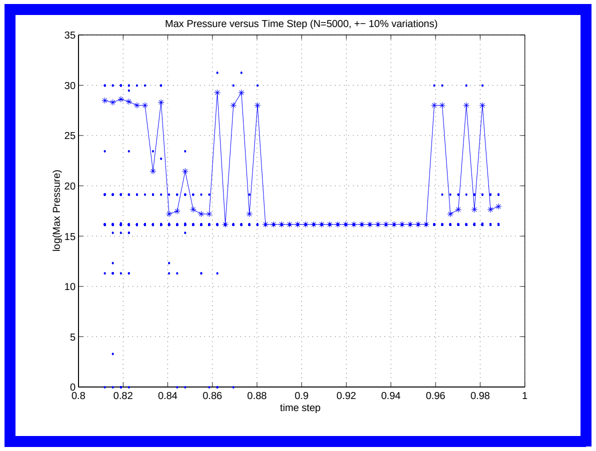


Mesh generation
DOE (sampling) and ALEGRA input
Processor assignments (MPI)
DDACE
"ALEGRA"
Interface
1, ..., N runs on 1, ..., M processors (MPI)
Returns status
Returns output data
Analysis (function approximation and response surfaces) - MARS; plotting (PGPLOT); etc
Inverse problem and optimization
Surrogate Objective Optimization Framework
PDS/PIO I/O library communication (Sturtevant, Christon, Heermann, Chen)

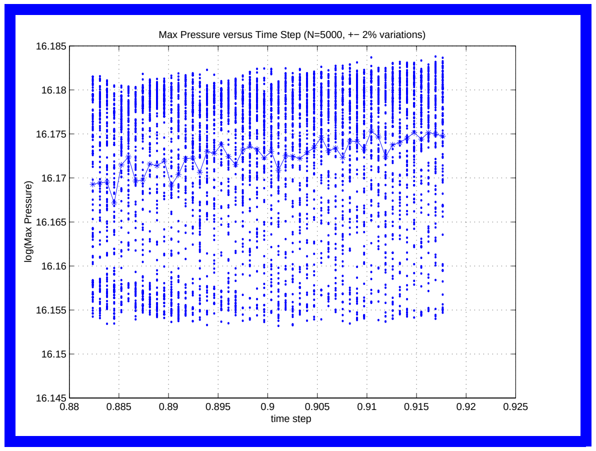
DDACE Application - strong shock in an ideal gas

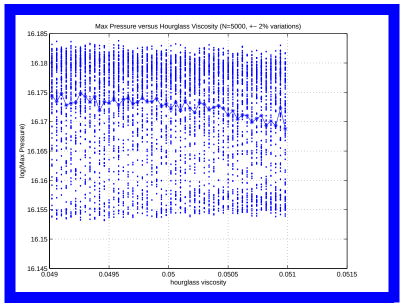


P_{max} vs dt

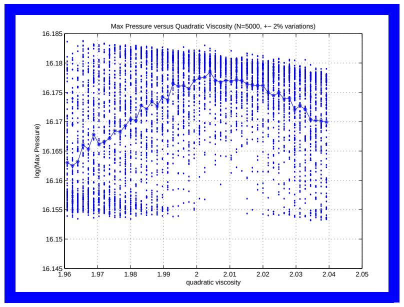


P_{max} vs dt

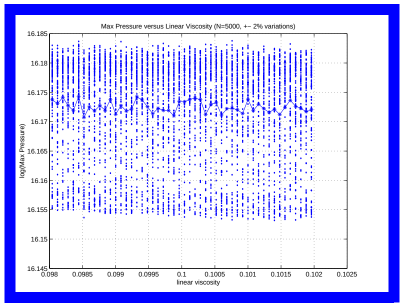




P_{max} vs hourglass



P_{max} vs quadratic



P_{max} vs linear

In conclusion:

"Summary: Computers Are Here To Stay. They Endanger Thought, Language, Science, and the Survival of Man. Like Any Other Dangerous Tool, They Should Be Put Under Strict Controls."

Clifford Truesdell, "The Computer: Ruin of Science, Threat to Man" in An Idiot’s Fugitive Essays on Science